

**成都远向电子科技有限公司产品说明书**

产品型号：YX-AI/DO-RS485-008

全部资料下载地址：http://ask.zstel.com:8090

技术支持服务电话：028-64267900

技术支持专员企业QQ：3183329475

官网网站：https://www.cdyxiot.com/

硬件/软件技术定制热线：19150158475 张工

## 目录

[目录 2](#_Toc81386694)

[一、产品概述 3](#_Toc81386695)

[1.1 概述 3](#_Toc81386696)

[1.2 性能特点 3](#_Toc81386697)

[1.3 技术参数 3](#_Toc81386698)

[二、外观尺寸 4](#_Toc81386699)

[2.1 产品外观 4](#_Toc81386700)

[2.2 产品尺寸图 4](#_Toc81386701)

[三、产品接线图、跳线、指示灯说明 6](#_Toc81386702)

[3.1 端子描述 6](#_Toc81386703)

[3.2 跳线 6](#_Toc81386704)

[3.3 LED指示灯 7](#_Toc81386705)

[3.4 接线示意图 7](#_Toc81386706)

[四、ModbusRTU通讯协议、组态软件软件说明 9](#_Toc81386707)

[4.1 通讯协议 9](#_Toc81386708)

[4.2 寄存器地址 9](#_Toc81386709)

[4.2 Modbus RTU功能码 10](#_Toc81386710)

[4.3 Modbus通讯实例 10](#_Toc81386711)

[五、软件操作 12](#_Toc81386712)

[5.1 配置软件 12](#_Toc81386713)

[5.2 配置基本参数 13](#_Toc81386714)

[5.3 DO继电器输出相关参数 13](#_Toc81386715)

[5.4 AI模拟量采集相关参数 13](#_Toc81386716)

[5.5 模拟量通道校准相关参数 14](#_Toc81386717)

[5.6 其他功能 15](#_Toc81386718)

[六、协议详解 21](#_Toc81386719)

[6.1 功能码描述 21](#_Toc81386720)

[6.2 错误码描述 27](#_Toc81386721)

[6.3 CRC校验算法 27](#_Toc81386722)

[七、更改记录 30](#_Toc81386723)

## 一、产品概述

### 概述

YX-AI/DO-RS485-008是一款工业级标准模拟量采集产品，共有 8个测量通道。每个通道均可以分别设置多种 量程；RS485 通讯接口使用标准 Modbus RTU 协议，符合工业标准。

### 性能特点

* 防死机硬件看门狗
* 5~35V带防反接、过压过流保护电源
* 8路模拟量电流输入0～20mA
* 8路模拟量电压输入0～10V、0～30V
* 8路NPN开关量信号输出
* 12位分辨率，0.1%精度ADC
* 高性能低功耗32位ARM嵌入式CPU
* 支持ModbusRTU从站协议
* 12路指示灯
* 带防雷、静电保护RS485通讯接口
* 工业温度范围，应对严苛现场环境
* 自定义线性模拟量数据转换

### 技术参数

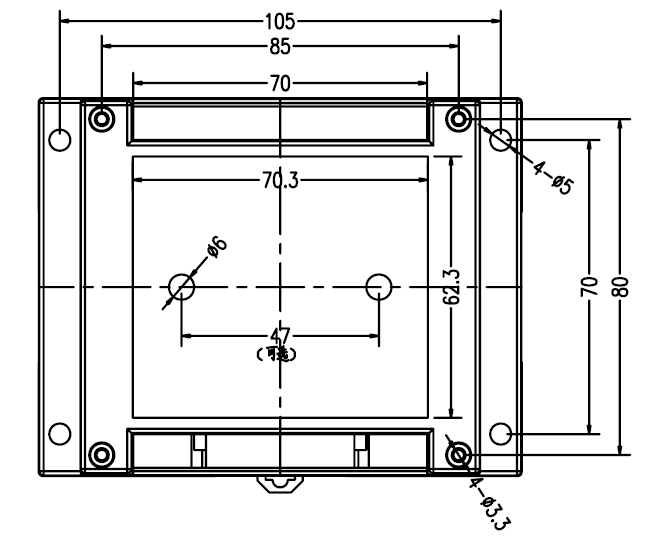
|  |  |  |
| --- | --- | --- |
| 模拟量接口 | AI | 8路单端 |
| AI分辨率 | 12bit |
| AI量程 | 0～10V、0～30V、0～20mA |
| 精度 | 0.1% |
| 采集速度 | 1KHz |
| AI输入阻抗 | 0～20mA ≤120Ω  0～10V/0～30V ≥ 10KΩ |
| 开关量接口 | DO | 8路单端 |
| 输出信号类型 | NPN |
| 输出电流电压 | Vout=VCC  额定最大负载 12V/0.5A |
| 通讯接口 | 通讯接口 | RS485 |
| 波特率 | 1200～115200bps |
| 数据格式 | 8N1,8E1,8O1,8N2,8E2,8O2 |
| 通讯协议 | ModbusRTU |
| 过压过流保护 | 45V |
| 电源参数 | 电源规格 | DC 5～36V |
| 功耗 | 12V-0.5w |
| 工作环境 | 工作温度、湿度 | -40℃～85℃，0%RH～95%RH |
| 其他 | 尺寸 | 82\*50\*32 |

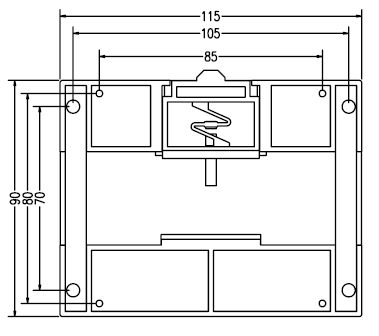
## 二、外观尺寸

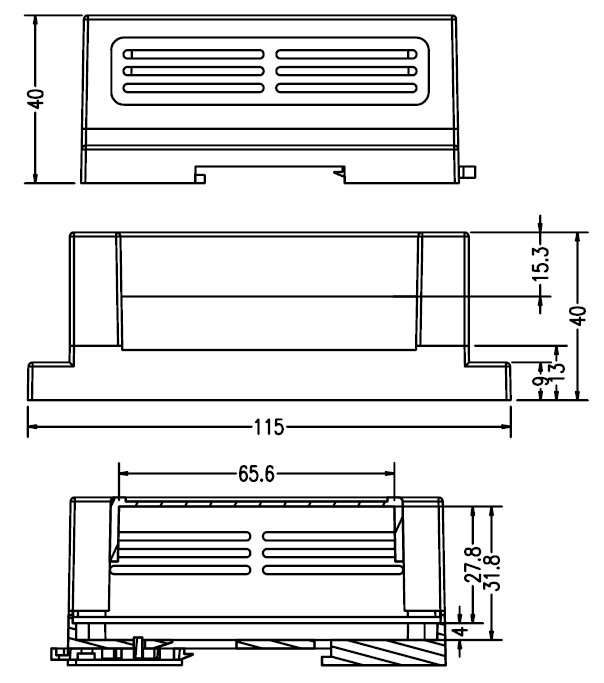
### 2.1 产品外观

### 2.2 产品尺寸图



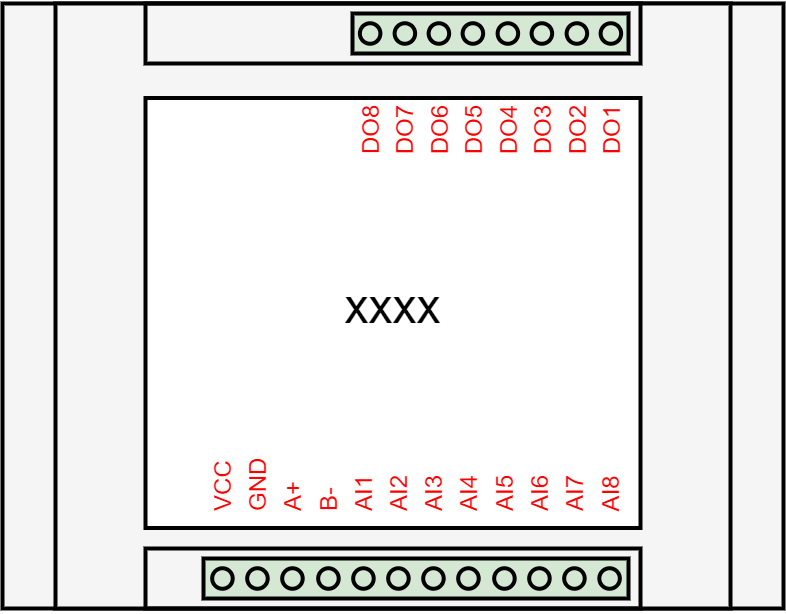






## 三、产品接线图、跳线、指示灯说明

### 3.1 端子描述

顶部8槽接线位：

OUT1：DO1输出端 OUT3：DO3输出端 OUT5：DO5输出端 OUT7：DO7输出端

OUT2：DO2输出端 OUT4：DO4输出端 OUT6：DO6输出端 OUT8：DO8输出端

底部12槽接线位：

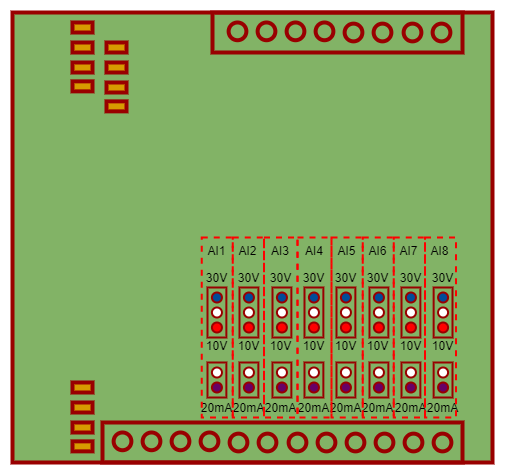
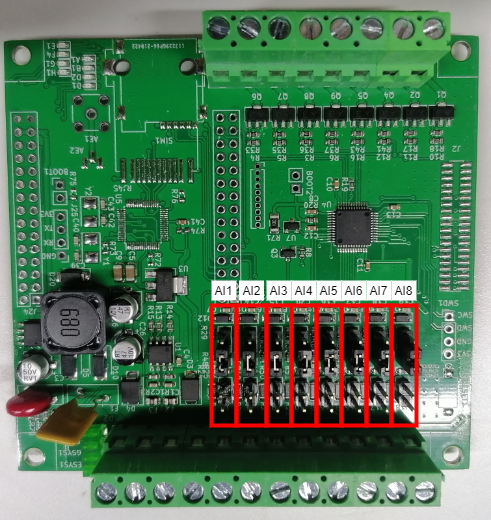
VDC： 电源正极端 AI1：模拟量输入端1 AI5：模拟量输入端5

GND： 电源负极端 AI2：模拟量输入端2 AI6：模拟量输入端6

A+： 485通讯线A AI3：模拟量输入端3 AI7：模拟量输入端7

B-： 485通讯线B AI4：模拟量输入端4 AI8：模拟量输入端8

### 3.2 跳线

电路板上有8个跳线座，通过跳线选择不同点量程，结合配置软件实现多量程功能：

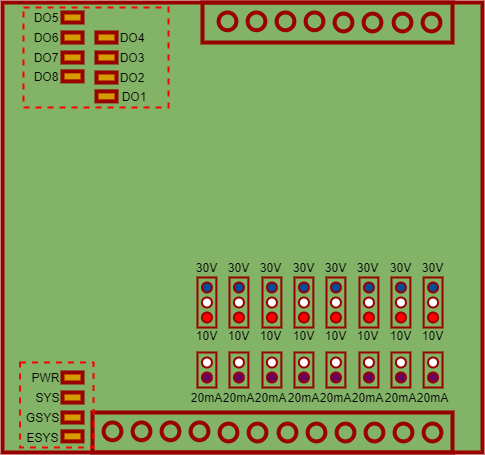
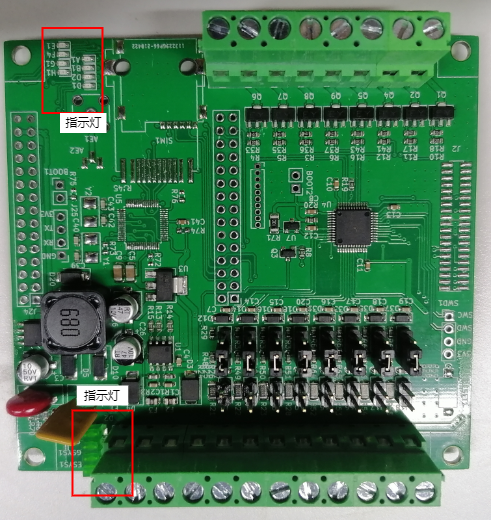
* 通道AIx：

0～10V量程： 选择红色侧2个跳线座

0～30V量程： 选择蓝色侧2个跳线座

0～20mA量程： 选择紫色侧1个跳线座

### 3.3 LED指示灯

12个LED指示灯：

顶部8个指示灯

* D1： DO1输出状态指示灯。 亮：DO有输出 灭：DO无输出
* D2： DO2输出状态指示灯。 亮：DO有输出 灭：DO无输出
* D3： DO3输出状态指示灯。 亮：DO有输出 灭：DO无输出
* D4： DO4输出状态指示灯。 亮：DO有输出 灭：DO无输出
* D5： DO5输出状态指示灯。 亮：DO有输出 灭：DO无输出
* D6： DO6输出状态指示灯。 亮：DO有输出 灭：DO无输出
* D7： DO7输出状态指示灯。 亮：DO有输出 灭：DO无输出
* D8： DO8输出状态指示灯。 亮：DO有输出 灭：DO无输出

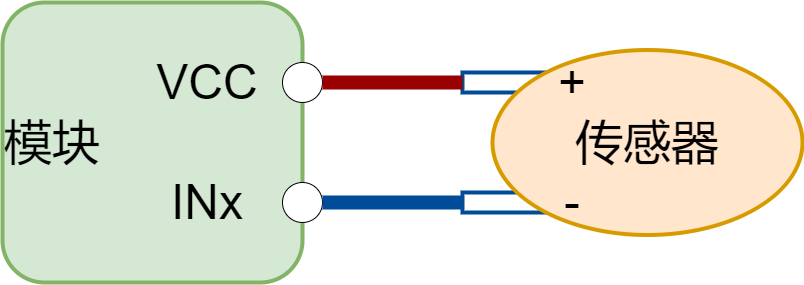
底部4个指示灯

* ESYS： 预留
* GSYS： 预留
* PWR： 正常供电时常亮
* SYS： 系统状态灯，正常运行时每秒闪烁一次

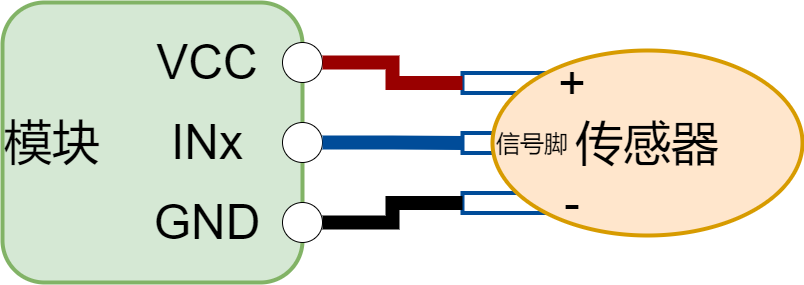
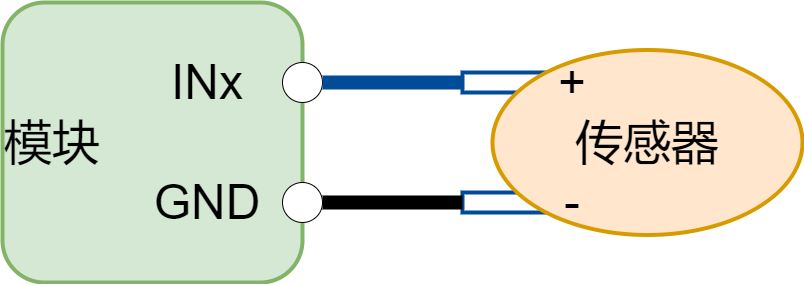
### 3.4 接线示意图

（1）模拟量输入接线

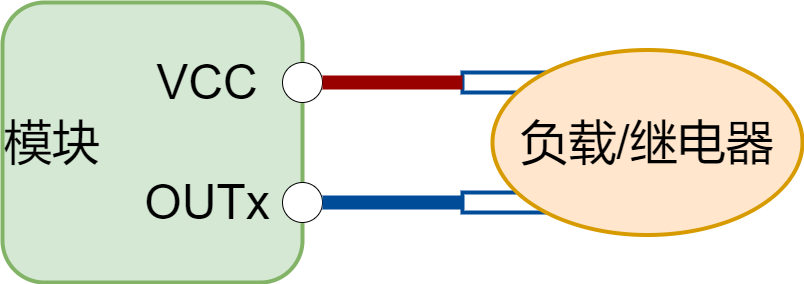
无源传感器（2线制）：



有源传感器（3线制）： 有源传感器（2/4线制）：

（2）开关量输出接线



## 四、ModbusRTU通讯协议、组态软件软件说明

### 4.1 通讯协议

本产品兼容标准Modbus RTU从站协议，能够支持标准Modbus RTU组态软件，详细内容介绍参考本文第六章内容

快速上手可参照“[4.3 Modbus通讯实例](#_5.3_Modbus通讯实例)”

### 4.2 寄存器地址

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 寄存器地址 | 名称 | 字节数 | 说明 | 备注 |
| 模拟量输入 | | | | |
| 0x0000(0) | AI1\_H | 2 | 模拟量通道 1 高 | 每个模拟量通道占 2 个 Modbus 寄存器，4 个字节，格式为浮点数，浮点数格式符合IEEE 754 标准  可参照4.3读取AI  0～10V: 0.0～10.0  0～30V: 0.0～30.0  0～20mA: 0.0～20.0 |
| 0x0001(1) | AI1\_L | 2 | 模拟量通道 1 低 |
| 0x0002(2) | AI2\_H | 2 | 模拟量通道 2 高 |
| 0x0003(3) | AI2\_L | 2 | 模拟量通道 2 低 |
| 0x0004(4) | AI3\_H | 2 | 模拟量通道 3 高 |
| 0x0005(5) | AI3\_L | 2 | 模拟量通道 3 低 |
| 0x0006(6) | AI4\_H | 2 | 模拟量通道 4 高 |
| 0x0007(7) | AI4\_L | 2 | 模拟量通道 4 低 |
| 0x0008(8) | AI5\_H | 2 | 模拟量通道 5 高 |
| 0x0009(9) | AI5\_L | 2 | 模拟量通道 5 低 |
| 0x000A(10) | AI6\_H | 2 | 模拟量通道 6 高 |
| 0x000B(11) | AI6\_L | 2 | 模拟量通道 6 低 |
| 0x000C(12) | AI7\_H | 2 | 模拟量通道 7 高 |
| 0x000D(13) | AI7\_L | 2 | 模拟量通道 7 低 |
| 0x000E(14) | AI8\_H | 2 | 模拟量通道 8 高 |
| 0x000F(15) | AI8\_L | 2 | 模拟量通道 8 低 |
| 0x0100(256) | AI1\_D | 2 | 模拟量通道1 | AI整数寄存器  0～20mA：0～2000  0～10V：0～1000  0～30V：0～3000 |
| 0x0101(257) | AI2\_D | 2 | 模拟量通道2 |
| 0x0100(258) | AI3\_D | 2 | 模拟量通道1 |
| 0x0101(259) | AI4\_D | 2 | 模拟量通道2 |
| 0x0100(260) | AI5\_D | 2 | 模拟量通道1 |
| 0x0101(261) | AI6\_D | 2 | 模拟量通道2 |
| 0x0100(262) | AI7\_D | 2 | 模拟量通道1 |
| 0x0101(263) | AI8\_D | 2 | 模拟量通道2 |
| 数字量输出 | | | | |
| 00014 | DO1 | 2 | 数字量通道1 | 0000 表示断开  0001 表示闭合 |
| 00015 | DO2 | 2 | 数字量通道2 |
| 00016 | DO3 | 2 | 数字量通道3 |
| 00017 | DO4 | 2 | 数字量通道4 |
| 00018 | DO5 | 2 | 数字量通道5 |
| 00019 | DO6 | 2 | 数字量通道6 |
| 0001A | DO7 | 2 | 数字量通道7 |
| 0001B | DO8 | 2 | 数字量通道8 |

### 4.2 Modbus RTU功能码

|  |  |  |
| --- | --- | --- |
| 功能码 | 操作 | 说明 |
| 01 | 读取单位DO状态 | Bit位表示DO输出状态 |
| 03 | 读取AI，DO寄存器值 | 读取AI，DO寄存器值 |
| 04 | 读取AI，DO寄存器值 | 读取AI，DO寄存器值 |
| 05 | 写单个DO | 0xFF00：闭合;0x0000：断开 |
| 06 | 写单个DO | 0x0001：闭合;0x0000：断开 |
| 0F | 写多个DO | 参照本文第六章内容 |
| 10 | 写多个DO | 参照本文第六章内容 |

详细讲解参照本文第六章内容

### 4.3 Modbus通讯实例

#### （1）读取DO1：

a．用01功能码读取DO1

发送：01 01 00 14 00 01 BD CE

接受：01 01 01 00 51 88

b．用03功能码读取DO1

发送：01 03 00 14 00 01 C4 0E

接受：01 03 02 00 00 B8 44

c．用04功能码读取DO1

发送：01 04 00 14 00 01 71 CE

接受：01 04 02 00 00 B9 30

#### （2）操作DO1：

a．用05功能码操作单个DO1

发送：01 05 00 14 FF 00 CC 3E

接受：01 05 00 14 FF 00 CC 3E

b．用06功能码操作单个DO1

发送：01 06 00 14 00 01 08 0E

接受：01 06 00 14 00 01 08 0E

c．用0F功能码操作多个DO1、DO2

发送：01 0F 00 14 00 02 01 03 AE 95

接受：01 0F 00 14 00 02 94 0E

d．用10功能码操作多个DO1、DO2

发送：01 10 00 14 00 02 04 00 01 00 01 63 50

接受：01 10 00 14 00 02 01 CC

#### （3）读取AI：

给定输入 4.96（40 9E E7 CF）

a．用03功能码读取浮点数AI1:

发送：01 03 00 00 00 02 C4 0B

接受：01 03 04 40 9E E7 CF 85 B9

4.96 IEE浮点数十六进制为（40 9E E7 CF）

b．用04功能码读取浮点数AI1：

发送：01 04 00 00 00 02 71 CB

接受：01 04 04 40 9E CE 1F 9A 02

C．用03功能码读取整数AI1：

发送：01 03 01 00 00 01 85 F6

接收：01 03 02 01 F0 B9 90

整数读出数值为 496（0x01F0）

d．用04功能码读取整数AI1：

发送：01 04 01 00 00 01 30 36

接收：01 04 02 01 F0 B8 E4

整数读出数值为 496（0x01F0）

## 五、软件操作

设备参数配置教程，结合《用户测试文档》即可对设备进行简单测试

### 5.1 配置软件

参数配置软件介绍：



5.1.1 配置软件包含有:

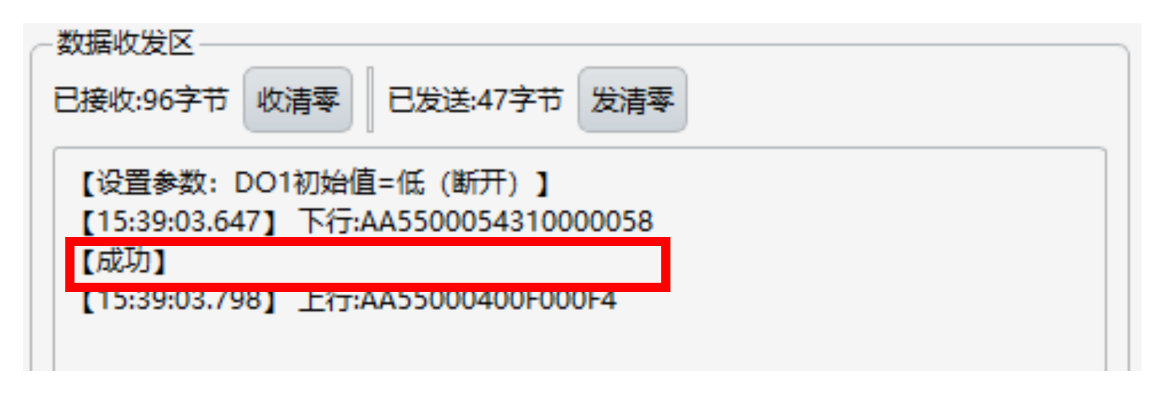
* 功能区: 包含有配置软件所支持功能,以及功能那个切换选项
* 参数配置主区域: 参数配置主要区域，参数项的读取、写入临时列表
* 串口/命令集区: 涉及模块的参数读、写、重启等操作
* 串口日志区: 命令集的操作日志

5.1.2参数配置准备：

1. 用USB-485工具连接设备到电脑
2. 在串口配置框内配置串口波特率、停止位、校验位、数据位;（默认波特率9600，数据位8，停止位1，校验位None）
3. 选择串口配置框子项“命令集”



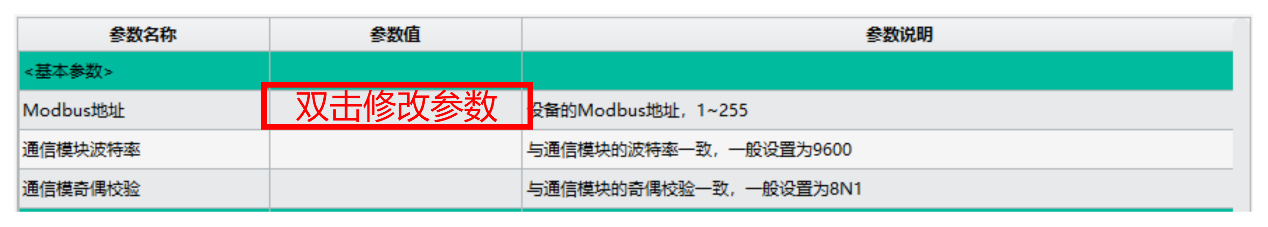
1. 点击“读取参数”命令按钮，读取设备参数（不同设备拥有不同指令集）
2. 双击对应参数项的“参数值”，然后对参数进行修改
3. 修改完参数后需要点击命令集里的“设置参数”，写入到模块中
4. 写入完成在日志区域会提示成功。



1. 通过点击“重启设备”按钮，重启模块设备使配置参数生效

### 5.2 配置基本参数

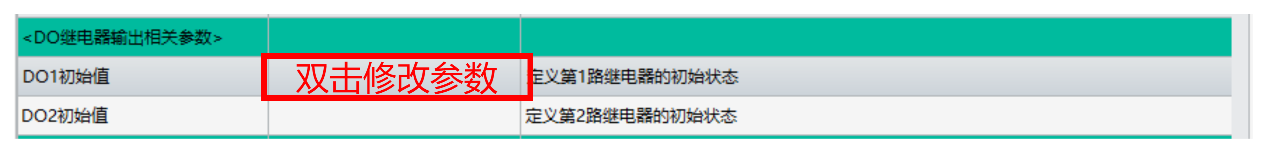
该系列参数涉及到对485通讯Modbus协议相关配置。



* Modbus地址：Modbus地址参数,可设置1～255
* 通讯模块波特率：设备485通讯波特率（波特率支持主流的波特率选项）
* 通讯模块就校验：设备485通讯奇偶校验位，可配置8N1,8E1,8O1...

### 5.3 DO继电器输出相关参数

部分产品包含有多个DO输出或者不包含有DO输出功能，具体请根据实际配置软件显示栏目进行配置。DO功能测试可以参考《用户测试文档》。

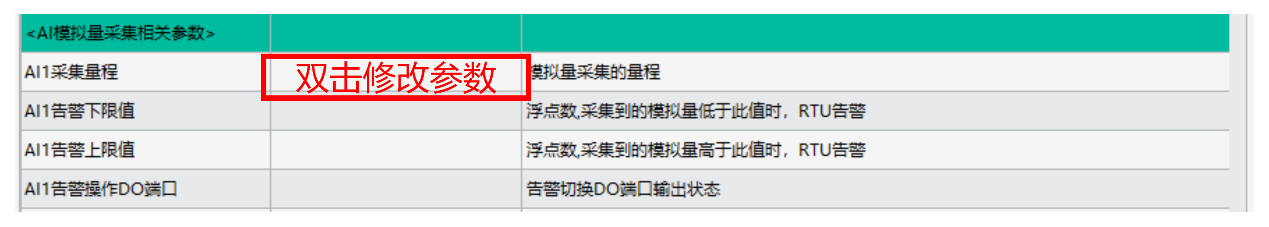


* DOx初始值：通过选择参数可配置开机上电后DO输出状态

默认为“等待上位操作”状态。

### 5.4 AI模拟量采集相关参数

本系列参数涉及到对AI量程、AI告警功能的配置。每个通道都包含如下的配置项。



* AIx采集量程：选择对应选项修改AI采集量程
* AIx告警下限值：采集数据低于此值时会触发AIx读数低于下限值告警事件，告警事件可用于DO，告警消息联动功能
* AIx告警上限值：采集数据低于此值时会触发AIx读数高于于下限值告警事件，告警事件可用于DO，告警消息联动功能
* AIx告警操作DO端口：当模拟量告警事件发生后，触发对DO的状态切换（由“低/断开”切换为“高/闭合”状态）

### 5.5 模拟量通道校准相关参数

本系列参数用于模拟量读数转换，RTU采集模块默认读取出来的读数值为采集的电流/电压值，接传感器时有事会用到将电流/电压值转换为实际传感器读数。则可以使用本系列参数实现读数转换功能。



* AIx比例值ratio: AIx数值转换公式中的比例值(详细使用参考后续说明)
* AIx偏移值offset: AIx数值转换公式中的偏移值(详细使用参考后续说明)

#### **5.5.1转换原理讲解**

模拟量读数转换公式如下（例：将电压转换为温度值）

AIx即转换后数值：

adc 为转换前读数值

ratio 为设定比例系数（初始为1.0）

offset 为设定位置系数（初始为0.0）

#### **5.5.2 校准实例**

本案例举例说明如何将一个4～20mA对应 -40℃～120℃的温度传感器输出的电流值转换为实际温度读数。

* 已知温度传感器 输出模拟量范围是 4～20mA，则将RTU采集模块量程设置为0/4～20mA量程范围。
* 计算需要的比例（ratio）、偏移（offset）三个参数值。

根据传感器参数可知

代入 ratio、offset计算公式中可得

ratio = [120 – (-40)] / (20 - 4) = 10

offset = {[20 \* (-40)] – [4 \* (120)]} / (20 - 40) = -80

* 根据将计算后的参数代入转换公式即可得

传感器输出模拟量4mA ---> 传感器温度读数 = 4\*10–80= -40℃

传感器输出模拟量20mA --> 传感器温度读数 = 20\*10–80= 120℃

### 5.6 其他功能

#### 5.6.1校准模拟量精度

当系统精度不够时，产品提供一个用户精度校准功能

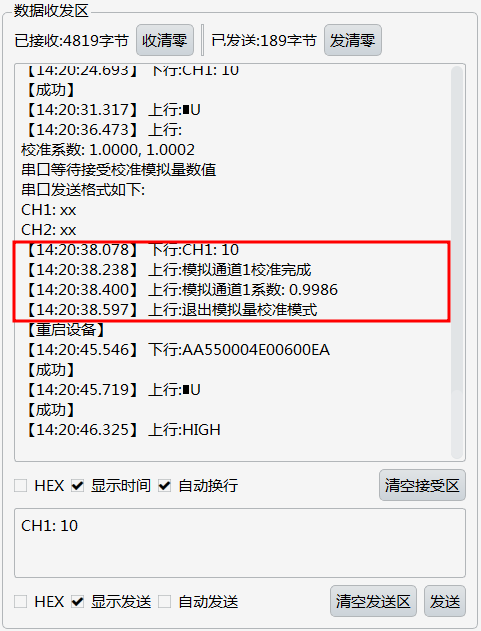
（1）设备串口连接上电脑，点击“命令集”中“校准模拟量精度”按钮进行模拟量校准



1. 取消勾选显示区“HEX”复选框，等到接受区提示如下内容时执行下一步



1. 给定模拟量CHx通道恒定基准信号（注意通道量程）
2. 发送通道基准信号格式 “CHx：xx”（例：“CH1：10”）注意取消勾选发送区“HEX”复选按钮
3. 等待设备提示校准消息



1. 重启设备

#### 5.6.2 AI-DO联动

注意: 部分产品不包含DO功能即无AI-DO联动功能，实际请根据所购买的模块进行测试。

1. 必要参数



* DOx初始值：配置正常空闲状态DOx状态
* AIx采集量程： 模拟量输入的量程范围
* AIx告警下限值：设置AIx最低触发值，若不用可设置为模拟量最小值以下（4～20mA，设置的值小于4mA即用不能达到）
* AIx告警上限值：设置AIx最高触发值，若不使用上限值可设置大于最大量程值即不会触发。

1. 实际案例

配置如下参数：

DO1初始值：低（断开）

AI1采集量程：电流0～20mA

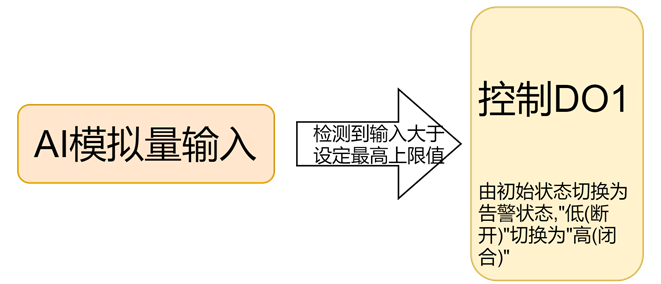
AI1告警下限值：1.0

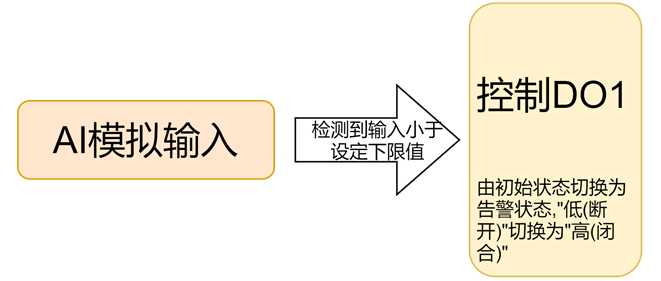
AI1告警上限值：15.0

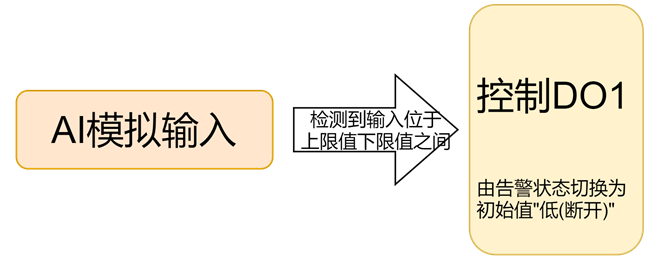
AI1告警操作DO端口：DO1



告警触发流程如下







#### 5.6.3 消息告警内容参数配置(无线版)

（1）必要参数

注：需要配合本公司远程模块,485有线版不开放该功能

* 安装地址： 主要用于报警通知使用（设备需包含有短信模块）
* 设备身份ID： 主要用于报警通知使用（设备需包含有短信模块）
* AIx告警周期: 用于重复发送告警消息
* AIx低于下限值告警短信内容： AI告警内容
* AIx恢复短信内容： AI告警内容
* AIx超出上限告警短信内容： AI告警内容



告警内容案例：

配置如下：

安装地址：“北厂区：”

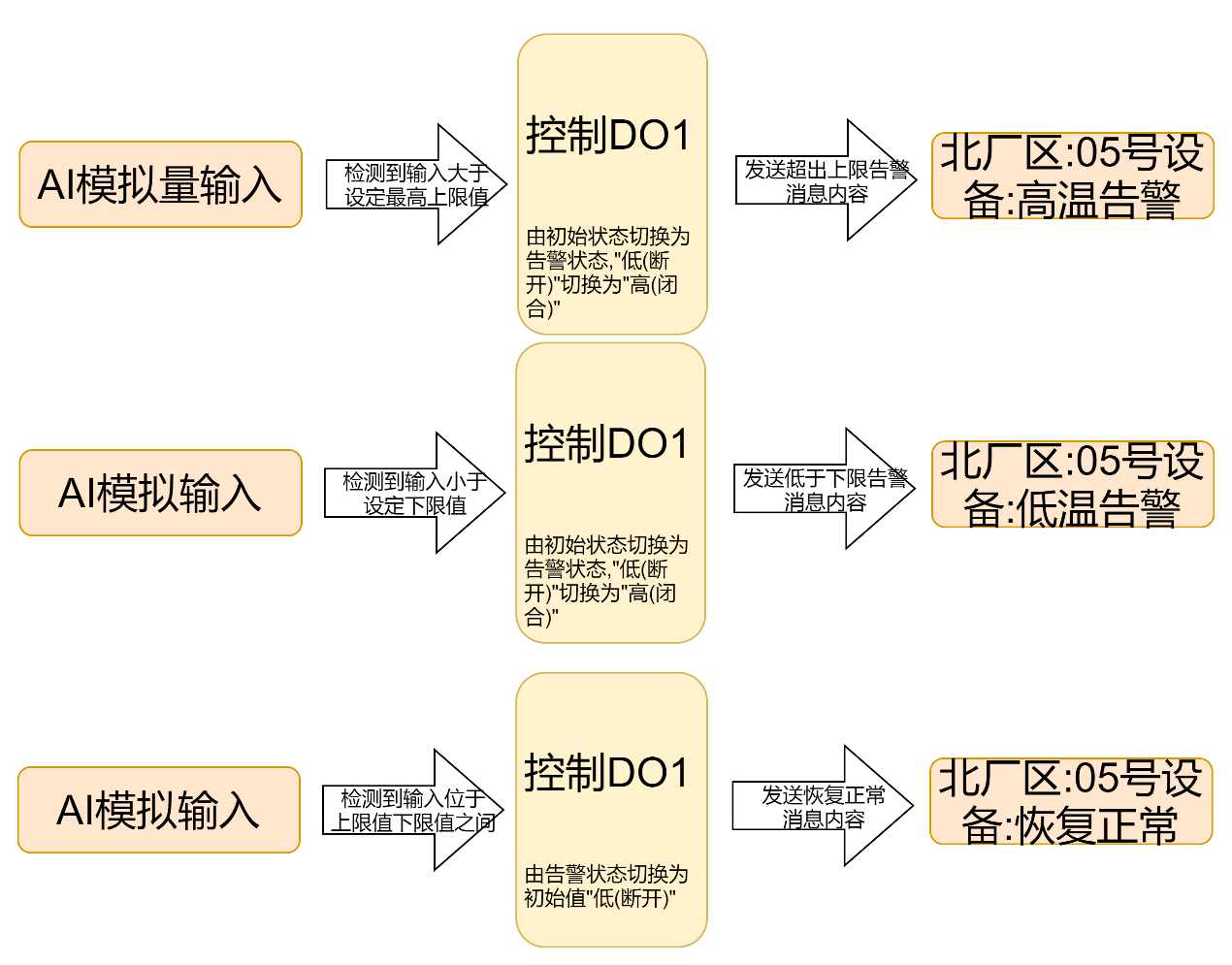
设备身份ID：“05号设备：”

AI1低于下限值短信内容：“低温警告”



其他告警短信内容格式与低于下限值格式一致（消息内容为空则不法送内容）

短信告警流程如下



#### 5.6.3 RTU数据可视化测试

参数配置软件包含有RTU数据可视化功能,可实现简单的模块功能测试。

操作步骤如下图：



* 选择“RTU可视化”界面
* 配置USB转485通讯参数并打开串口
* 设置Modbus子站地址为默认“1”
* 点选“自动读取”复选框

注意: 此时界面左侧仪表盘会根据模拟量信号源的输出显示对应电流读数，右侧会有上下行通讯报文。**若无上行数据请检查USB-485转换器是否正常工作，或尝试模块485 AB之间接120欧电阻。**

详细的测试演示可参考“用户测试文档”

## 六、协议详解

|  |  |  |  |
| --- | --- | --- | --- |
| 地址域 | 功能码 | 数据 | 差错检验 |

Modbus使用“big-Endian”（大端模式）表示地址和数据项，这就意味着当发射多个字节时，首先发送最高字节。

例如：寄存器地址为0x0014，首先发送的是0x00，然后才是0x14。

一个正常的Modbus响应：响应功能码=请求功能码。

一个Modbus的异常响应：响应功能码=请求功能码+0x80，提供一个异常码来指示差错原因。

### 6.1 功能码描述

#### 6.1.1 01读线圈

可以使用此功能码读取继电器DOx的状态。

请求PDU详细说明了起始地址，即指定第一个线圈的地址和线圈数量，从零开始寻址线圈，因此寻址线圈1-N为0-(N-1)。

响应PDU中N个字节的线圈状态的每一个bit位代表一个线圈的状态，状态1=ON, 0=OFF。第一个字节的最低位LSB代表第0号线圈的状态（即起始地址指定的线圈号为0号线圈），其他线圈依次类推，一直到这个字节的最高位MSB为止，并且后续字节中都是由低到高代表连续的各线圈状态。

如果线圈数量不是8的倍数，将用零填充剩余最后数据字节中的剩余比特，字节数量域说明了数据的完整字节数。

请求PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x01 |
| 起始地址 | 2个字节 | 0x0014至0x0015 |
| 线圈数量 | 2个字节 | n(1至n-1) |
| CRC校验 | 2个字节 |  |

注: 线圈状态的字节数N=线圈数量n/8, 如果余数不等于0，则N=n/8+1

错误响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x81 (请求功能码+0x80) |
| 异常码 | 1个字节 | 0x01或0x02或0x03或0x04 |
| CRC校验 | 2个字节 |  |

这是一个读离散量DO1的实例

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 地址 | 01 | 地址 | 01 |
| 功能码 | 01 | 功能码 | 01 |
| 起始地址高H | 00 | 字节数 | 01 |
| 起始地址低L | 14 | DO1-DO4状态 | 01 |
| 线圈数量高H | 00 | CRC校验高H | 90 |
| 线圈数量低L | 01 | CRC校验低L | 48 |
| CRC校验高H | BD |  |  |
| CRC校验低L | CE |  |  |

**发送：010100140001BDCE RTU响应：010101019048**

DO1的状态字节为0D，二进制00000001，DO1是这个字节的LSB(第0位)为1表示闭合，其他DOx是第(x-1)位为0表示断开，用0填充未使用位。

#### 6.1.2 03读保持寄存器/04读输入寄存器

使用该功能码可以读取所有寄存器包括AIx、DOx、DIx的状态。

请求PDU详细说明了起始寄存器地址和寄存器数量，从0开始寻址寄存器，因此寻址寄存器1-N为0-(N-1)。

响应报文中的寄存器数据每个寄存器有2个字节，对于每一个寄存器，第一个字节代表寄存器值的高位，第二个字节代表寄存器值的低位。字节数为寄存器数量乘以2。对于AI，一个通道占用2个寄存器，4个字节的值使用浮点数表示，对于DOx，2个字节的值0000代表继电器断开，0001代表继电器闭合,对于DIx,2个字节的值0000代表开关量无输入,0001代表有输入。

请求PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x03或04 |
| 起始地址 | 2个字节 | 0x0000至0x0017 |
| 寄存器数量 | 2个字节 | n(1至N) |
| CRC校验 | 2个字节 |  |

响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x03或0x04 |
| 字节数 | 1个字节 | N=2\*n |
| 寄存器值 | N个字节 | N=2\*n, n为寄存器数量 |
| CRC校验 | 2个字节 |  |

错误响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x83或0x84 (请求功能码+0x80) |
| 异常码 | 1个字节 | 0x01或0x02或0x03或0x04 |
| CRC校验 | 2个字节 |  |

这是一个读模拟量输入AI1的实例

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 地址 | 01 | 地址 | 01 |
| 功能码 | 03 | 功能码 | 03 |
| 起始地址高H | 00 | 字节数 | 04 |
| 起始地址低L | 00 | AI1值 | 4个字节 |
| 寄存器数量高H | 00 | CRC校验高H |  |
| 寄存器数量低L | 02 | CRC校验低L |  |
| CRC校验高H | C4 |  |  |
| CRC校验低L | 0B |  |  |
|  |  |  |  |

**发送：010300000002C40B RTU响应:0103044019999AD40F**

#### 6.1.3 05写单个线圈

可以使用该功能码写单个继电器DOx为断开或闭合

请求数据域中的常量说明请求的ON/OFF状态，十六进制值0xFF00请求输出为ON(闭合)，十六进制值0x0000请求输出为OFF(断开)，其他所有值都是非法的，对输出不起作用，RTU返回错误响应。

请求域中的输出地址规定了要写入线圈的地址。

正常响应是请求的应答，在写入线圈状态后返回这个正常响应。

请求PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x05 |
| 输出地址 | 2个字节 | 0x0014至0x0015 |
| 输出值 | 2个字节 | 0x0000或0xFF00 |
| CRC校验 | 2个字节 |  |

响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x05 |
| 输出地址 | 2个字节 | 0x0014至0x0015 |
| 输出值 | 2个字节 | 0x0000或0xFF00 |
| CRC校验 | 2个字节 |  |

错误响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x85 (请求功能码+0x80) |
| 异常码 | 1个字节 | 0x01或0x02或0x03或0x04 |
| CRC校验 | 2个字节 |  |

这是一个请求写线圈DO1为ON(闭合)的实例

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 地址 | 01 | 地址 | 01 |
| 功能码 | 05 | 功能码 | 05 |
| 输出地址高H | 00 | 输出地址高H | 00 |
| 输出地址低L | 14 | 输出地址低L | 14 |
| 输出值高H | FF | 输出值高H | FF |
| 输出值低L | 00 | 输出值低L | 00 |
| CRC校验高H | CC | CRC校验高H | CC |
| CRC校验低L | 3E | CRC校验低L | 3E |

**发送：01050014FF00CC3E RTU响应：01050014FF00CC3E**

#### 6.1.4 06写单个寄存器

可以使用该功能码写单个继电器DOx为断开或闭合。

请求数据域中的寄存器值说明请求的ON/OFF状态，十六进制值0001请求输出为ON(闭合)，十六进制值0x0000请求输出为OFF(断开)。

请求域中的寄存器地址规定了要写入线圈的地址。

正常响应是请求的应答，在写入线圈状态后返回这个正常响应。

请求PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x06 |
| 寄存器地址 | 2个字节 | 0x0014至0x0015 |
| 寄存器值 | 2个字节 | 0x0000至0xFFFF |
| CRC校验 | 2个字节 |  |

响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x06 |
| 寄存器地址 | 2个字节 | 0x0014至0x0015 |
| 寄存器值 | 2个字节 | 0x0000至0xFFFF |
| CRC校验 | 2个字节 |  |

错误响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x86 (请求功能码+0x80) |
| 异常码 | 1个字节 | 0x01或0x02或0x03或0x04 |
| CRC校验 | 2个字节 |  |

这是一个请求写线圈DO1为ON(闭合)的实例

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 地址 | 01 | 地址 | 01 |
| 功能码 | 06 | 功能码 | 06 |
| 寄存器地址高H | 00 | 寄存器地址高H | 00 |
| 寄存器地址低L | 14 | 寄存器地址低L | 14 |
| 寄存器值高H | 00 | 寄存器值高H | 00 |
| 寄存器值低L | 01 | 寄存器值低L | 01 |
| CRC校验高H | 08 | CRC校验高H | 08 |
| CRC校验低L | 0E | CRC校验低L | 0E |

**发送：010600140001080E RTU响应：010600140001080**E

#### 6.1.5 0F写多个线圈

可以使用此功能码写多个继电器DOx为断开或闭合。

请求PDU详细说明了起始地址，即指定第一个线圈的地址和线圈数量，从零开始寻址线圈，因此寻址线圈1-N为0-(N-1)。

请求数据域中的内容说明了被请求的ON/OFF状态，域比特位中的逻辑“1”请求相应输出为ON, 域比特位中的逻辑“0”请求相应输出为OFF。从数据域中第一个字节的bit0开始到bit7，然后到第二个字节的bit0，依次表示第一个线圈到第n个线圈的ON/OFF值。

正常响应返回功能码、起始地址和线圈数量。

请求PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x0F |
| 起始地址 | 2个字节 | 0x0014至0x0015 |
| 线圈数量 | 2个字节 | n(1至N) |
| 字节数 | 1个字节 | N=n/8, 或N=n/8+1 |
| 输出值 | N个字节 |  |
| CRC校验 | 2个字节 |  |

注: 线圈输出字节数N=线圈数量n/8, 如果余数不等于0，则N=n/8+1

响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x0F |
| 起始地址 | 2个字节 | 0x0014至0x0015 |
| 线圈数量 | 2个字节 | n(1至2) |
| CRC校验 | 2个字节 |  |

错误响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x8F (请求功能码+0x80) |
| 异常码 | 1个字节 | 0x01或0x02或0x03或0x04 |
| CRC校验 | 2个字节 |  |

这是一个请求从线圈DO1开始写入1个线圈的实例

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 地址 | 01 | 地址 | 01 |
| 功能码 | 0F | 功能码 | 0F |
| 起始地址高H | 00 | 起始地址高H | 00 |
| 起始地址低L | 14 | 起始地址低L | 14 |
| 线圈数量高H | 00 | 线圈数量高H | 00 |
| 线圈数量低L | 01 | 线圈数量低L | 01 |
| 字节数 | 01 | CRC校验高H | D4 |
| 输出值 | 01 | CRC校验低L | 0F |
| CRC校验高H | DF |  |  |
| CRC校验低L | 54 |  |  |

**发送：010F0014000201012F51 RTU响应：010F00140001D40F**

DO1的输出值为01，二进制00000001，DO1是这个字节的LSB(第0位)为0表示断开，DOx是第(x-1)位为1表示闭合，用0填充剩余未使用位。

#### 6.1.6 10写多个寄存器

使用该功能码可以写连续寄存器DOx的状态。

请求PDU详细说明了起始寄存器地址、寄存器数量、字节数和寄存器值，从零开始寻址寄存器，因此寻址寄存器1-N为0-(N-1)。

寄存器数据中每个寄存器有2个字节，对于每一个寄存器，第一个字节代表寄存器值的高位，第二个字节代表寄存器值的低位。字节数为寄存器数量乘以2，2个字节的值0000代表继电器断开，0001代表继电器闭合。

正常响应返回功能码、起始地址和被写入寄存器的数量。

请求PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x10 |
| 起始地址 | 2个字节 | 0x0014至0x0015 |
| 寄存器数量 | 2个字节 | n(1至N) |
| 字节数 | 1个字节 | N=2\*n |
| 寄存器值 | N个字节 | N=2\*n, n为寄存器数量 |
| CRC校验 | 2个字节 |  |

响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x10 |
| 起始地址 | 2个字节 | 0x0014至0x0015 |
| 寄存器数量 | 2个字节 | n(1至2) |
| CRC校验 | 2个字节 |  |

错误响应PDU

|  |  |  |
| --- | --- | --- |
| 地址 | 1个字节 |  |
| 功能码 | 1个字节 | 0x90 (请求功能码+0x80) |
| 异常码 | 1个字节 | 0x01或0x02或0x03或0x04 |
| CRC校验 | 2个字节 |  |

这是一个控制继电器DOx的实例

|  |  |  |  |
| --- | --- | --- | --- |
| 请求 | | 响应 | |
| 地址 | 01 | 地址 | 01 |
| 功能码 | 10 | 功能码 | 10 |
| 起始地址高H | 00 | 起始地址高H | 00 |
| 起始地址低L | 14 | 起始地址低L | 14 |
| 寄存器数量高H | 00 | 寄存器数量高H | 00 |
| 寄存器数量低L | 01 | 寄存器数量低L | 01 |
| 字节数 | 02 | CRC校验高H | 41 |
| DO1寄存器值高H | 00 | CRC校验低L | CD |
| DO1寄存器值高L | 01 |  |  |
| CRC校验高H | 64 |  |  |
| CRC校验低L | 84 |  |  |

**发送：0110001400010200016484 RTU响应：01100014000141CD**

DO1寄存器值为0001表示闭合

### 6.2 错误码描述

错误码含义：当DTU收到错误的Modbus指令时，会返回功能码为请求功能码+0x80，紧随着一个字节的错误码代表出错原因。

错误码01：表示不支持的功能码，众山DTU支持上述8种功能码，除此之外的功能码都会返回错误码为01的错误。

错误码02：表示起始地址不存在或者起始地址加上寄存器数量后的地址不存在。总的来说表示访问的寄存器不存在。

错误码03：表示寄存器数量不符合规范或者寄存器值非法。

错误码04：表示读写寄存器错误。

### 6.3 CRC校验算法

CRC即[循环冗余校验码](https://baike.baidu.com/item/%E5%BE%AA%E7%8E%AF%E5%86%97%E4%BD%99%E6%A0%A1%E9%AA%8C%E7%A0%81)（Cyclic Redundancy Check ）：是数据通信领域中最常用的一种查错校验码，其特征是信息字段和校验字段的长度可以任意选定。循环冗余检查（CRC）是一种数据传输检错功能，对数据进行多项式计算，并将得到的结果附在帧的后面，接收设备也执行类似的算法，以保证数据传输的正确性和完整性。

ModbusRTU的CRC16计算初值: 0xFFFF

ModbusRTU的CRC16计算多项式0xA001 (二进制:1010 0000 0000 0001)

附CRC校验算法代码：

uint16\_t mb\_crc( uint8\_t\* snd, uint16\_t num )

{

    uint8\_t CRC\_Lb, CRC\_Hb;

    uint8\_t CRC\_L, CRC\_H;

    uint16\_t crc;

    CRC\_H = 0xFF;

    CRC\_L = 0xFF;

    for ( uint16\_t i = 0; i < num; i++ ) {

        CRC\_L = CRC\_L ^ snd[ i ];

        for ( uint16\_t j = 0; j < 8; j++ ) {

            CRC\_Lb = CRC\_L;

            if ( ( CRC\_L & 1 ) == 1 ) {

                CRC\_L  = ( CRC\_L - 1 ) / 2;

                CRC\_Lb = CRC\_L;

                CRC\_Hb = CRC\_H;

                if ( ( CRC\_H & 1 ) == 1 ) {

                    CRC\_L  = CRC\_L + 128;

                    CRC\_Lb = CRC\_L;

                    CRC\_H  = ( CRC\_H - 1 ) / 2;

                    CRC\_Hb = CRC\_H;

                } else {

                    CRC\_H  = CRC\_H / 2;

                    CRC\_Hb = CRC\_H;

                }

                CRC\_L  = CRC\_L ^ 1;

                CRC\_Lb = CRC\_L;

                CRC\_H  = CRC\_H ^ 0xA0;

                CRC\_Hb = CRC\_H;

            } else {

                CRC\_L  = CRC\_L / 2;

                CRC\_Lb = CRC\_L;

                CRC\_Hb = CRC\_H;

                if ( ( CRC\_H & 1 ) == 1 ) {

                    CRC\_L  = CRC\_L + 128;

                    CRC\_Lb = CRC\_L;

                    CRC\_H  = ( CRC\_H - 1 ) / 2;

                    CRC\_Hb = CRC\_H;

                } else {

                    CRC\_H  = CRC\_H / 2;

                    CRC\_Hb = CRC\_H;

                }

            }

        }

    }

    crc   = CRC\_L;

    crc <<= 8;

    crc  |= CRC\_H;

    return crc;

}

## 七、更改记录

v1.6

\* 新增AI整数段寄存器读取说明以及示例

v1.5

\* 更新有源、无源接线方式

v1.4

\* 更新外壳尺寸图

\* 更新电池检测接线方式

v1.3

\* 接线图新增三线制

v1.2

\* 修改接线图，调整文字字体

v1.1

\* 修改3.1接线图，3.2跳线图，3.3LED指示灯配图

v1.0

\* 第一版编写